DESCRIPTION AMENDMENTS

Rewrite the paragraph beginning on page 8, line 18, to read as follows:

Figure 2 is a block diagram illustrating an exemplary deployment of a network connection device 12 including the virtual machine 10 that accesses a set of classification rules 18 utilized to make traffic classification decisions. These classification rules 18 may be as simple or as complex as required to make a classification, and may define a "signature" of a particular type of network traffic in order to make a classification. For the purposes of the present location, the term "signature" shall be taken ~~to the~~ to be information pertaining to network traffic, whether extracted from the network traffic itself or not, ~~that the~~ that may be utilized to characterize or classify network traffic. Within the exemplary deployment shown in Figure 2, the virtual machine 10 is shown to receive network traffic from a number of 10baseT network connections via a number of ingress virtual interfaces 24, and to output classified network traffic via a number of egress virtual interfaces 26 to an ATM or ADSL network connection. In one embodiment, the virtual interfaces 24 and 26 may constitute a physical port and/or a virtual channel. Network traffic entering one of the ingress virtual interfaces 24 is operationally classified by the virtual machine 10 utilizing the classification rules 18. A packet, frame or cell is then routed, switched or bridged to an appropriate egress virtual interface 26, as defined by the classification rules 18.

Rewrite the paragraph beginning on page 12, line 7, to read as follows:

The signature 31 of a packet 29 is utilized by the classifier 14 to differentiate the packet 29 from other dissimilar packets. As stated above, sequences of packets (or other network traffic units) bearing ~~in the same~~ the same signature are termed "flows". A flow is said to be instantiated when the classifier 14 recognizes a packet 29 bearing the flow's signature, and persists until the amount of time between packets 29 bearing the flow's signature exceeds a particular amount of time (e.g., a flow's Interval Timeout).

Rewrite the paragraph beginning on page 15, line 6, to read as follows:

The transmit code point field, if specified, includes a value that will become a so-called transmit ?behavior code point? for an outgoing packet. The behavior code point is a value that indicates how the virtual machine 10 should forward a flow (i.e., it specifies algorithms that will be used to queue and forward the packet, etc.). Packet forwarding processing is protocol specific, so the behavior code point is a normalization of the semantics associated with packet forwarding. Once a forwarding decision is made in a packet, an egress virtual interface 26 will map this value into ~~it's own pier-to-pier protocol proprietary~~ its own proprietary peer-to-peer protocol for transmission.

Rewrite the paragraph beginning on page 15, line 14, to read as follows:

Regarding the reciprocal flow field, a flow can be configured to identify its reciprocal flow (i.e., any traffic in a reverse direction of the flow, which is generated as a result of that flow). This is depicted in Figure 6, where transactions 2 and 3 occur as a direct consequence of transaction 1. If a virtual interface is not configured to bind it?s reciprocal flow, the virtual machine 10 may identify transaction 2 and 3 as two flows (e.g., A.B flow with a count of 1 packet and a B.A flow with a flow of 2 packets). However, if the virtual interface is configured to bind ~~it's reciprocal~~ its reciprocal flow, the virtual machine 10 will recognize just a single flow (e.g., an A.B flow with a count of 3 packets).

Rewrite the paragraph beginning on page 16, line 6, to read as follows:

When the virtual machine 10 switches a packet to an egress virtual interface 26, the flow class to which the relevant packet belongs provides a transmit code point (e.g., the behavior code point discussed above), which specifies the transmission requirements of the relevant flow class. Each virtual interface is created to support a specific network topology, and to ~~specify how~~ specify how to map a packet to and from the external network. Specifically, each virtual interface includes configuration to set the type of underlying

physical interface (e.g., Ethernet, VDSL, ADSL, etc.), assign a driver instance (i.e., the realization of the physical layer), assign the label space of the physical layer that the virtual interface can use, set the type of virtual interface (e.g., Ethernet, RFC1483, PPPoverL2TP, etc.), enable disable DHCP, assign a MAC address, assign an IP address and subnet mask (when routing), enable and disable IP multicasting, enable and disable broadcasting to other virtual interfaces of a particular type, enable and disable Network Address Translation, and enable and disable Spanning Tree and set state (e.g., blocking, listening, forwarding, etc.) priority and cost.

Rewrite the paragraph beginning on page 17, line 10, to read as follows:

At block 44, a packet 29 is received at an ingress virtual interface 24 (e.g., ~~via a~~ via an Ethernet port or via a PCI bus). The packet 29 is then IP routed to the virtual machine 10 at block 46. At block 48, the signature, as described above, for the packet 29 is determined. At block 50, a policy to be applied in processing of the packet 29 is identified by utilizing the signature to perform a lookup on the policy and/or flow class tables 30 and 36.

Rewrite the paragraph beginning on page 18, line 7, to read as follows:

Component parameters are often dependent on each other, and may be mutually exclusive. Correct configuration of a network device requires careful consideration of these dependencies. Network management devices typically allow for the setting of individual component parameters, but do not enforce a net result of a series of discrete configuration operations. This may be due to the large amount of resources required in both the managing and managed devices to perform such a task. The above problem of configuring component parameters, which may be dependent upon each other, is becoming more prevalent as network devices are becoming smaller, more numerous, more functional, low cost, and more mission critical. Specifically, network devices are being increasingly deployed (some in mission critical applications), and network administration is becoming an increasing expense for organizations. The volume deployment of broadband services is contributing towards the ~~exasperation~~

exacerbation of the above-identified problem.

Rewrite the paragraph beginning on page 19, line 7, to read as follows:

Figure 9 is a block diagram providing a high level diagrammatic representation of the operation of a virtual machine compiler 60, according to an exemplary embodiment of the present invention. The virtual machine compiler 60 is shown to receive as inputs: (1) an operations file 62 that describes operations supported by components of a particular network device (i.e., component behavior) and constraint definitions, and (2) a ~~rule file~~ rule file 64 that specifies behavioral requirements of a specific network device. In one embodiment, these behavioral requirements may be specified as a textual representation in the form of a decision tree.

Rewrite the paragraph beginning on page 20, line 6, to read as follows:

The virtual machine compiler 60, in one embodiment, presents a model to a rule designer that consists of a number of abstract data processes and contexts, as illustrated in Figure 10. Specifically, Figure 10 illustrates the rule program 66 as conceptually comprising a number of rules 68 (i.e., instruction sequences) that are utilized to bind process behavior definitions, conveniently labeled operations 70, ~~to contextual sed~~ to contextualized sets of data, conveniently labeled registers 72. It will be appreciated that because a specific network connection device 12 may be constituted by a number of smaller components, the overall process and context for a network connection device 12 may ~~similarly viewed~~ similarly be viewed as constituting a number of corresponding components. As illustrated in Figure 10, each component (e.g., the TCP protocol or an ATM device driver) that wishes to contribute to a process (e.g., an abstract entity such as a data plane or the management plane) can operate, via a rule 68 class on a new or existing register 72.

Rewrite the paragraph beginning on page 20, line 19, to read as follows:

A particular component may together itself as multiple processes. For example, a component TCP may provide operations in both a data

plane ~~process,~~ process and a management plane process.

Rewrite the paragraph beginning on page 20, line 22, to read as follows:

A rule 68 is declared to be for a specific process 73, hook 74 and context 75, and the virtual machine compiler 60 operates to insure that all components and operations used in a specific rule 68 are compatible with that declaration. A hook 74 may be ~~regarded a~~ regarded as a location within a process to which a rule 68 may be addressed. Once a rule program 66 is written and tested, it may completely describe the behavior of a network connection device 12.

Rewrite the paragraph beginning on page 22, line 7, to read as follows:

An exemplary implementation of the virtual machine 10 may be broken down into a number of discrete and re-useable software parts, termed components, each of which ~~has as section~~ has a section within the operations file 62 ~~that described~~ that describes the operations supported by the respective component. A product model may be viewed as a specific instance of a virtual machine 10, which has a defined set of components. The virtual machine 10 described by a product model is only capable of executing the operations of its constituent components. Each component is assigned a global identity, and has its own operations namespace. At run time, the implementation of each component registers its operation with the virtual machine compiler 60. When a new rule is introduced into a network connection device 12 (e.g., via a network management or from memory), the virtual machine compiler 60 checks for consistency between a new rule and its registered implementation. An assigned identifier between 1 and 1216 ? 1 may identify components.

Rewrite the paragraph beginning on page 22, line 20, to read as follows:

Referring again to Figure 10, rules 68 in the rule program 66 are associated with abstract entities created by the virtual machine 10. These abstract entities are defined in terms of their behavior and their data. A particular process 73 uniquely identifies a particular behavior, and a context 75 uniquely identifies a particular data

environment. A process 73 and a context 75 required for correct operation of a rule program 66 are coded into an instruction sequence of the relevant rule program 66. The virtual machine 10 checks that the registered ~~implementations supports~~ implementations support the same process 73 and context 75 as required by a specific rule 68. The grammar of an exemplary operations file 62 is provided below:

Rewrite the paragraph beginning on page 24, line 19, to read as follows:

Operationally, the virtual machine compiler 60 insures that a rule program 66 does not execute a predicate operation after an action operation has been executed, because the change of systems implied by the action precludes any backtracking. A monitor operation (not shown) may change the state of a network connection device 12, as long as it does so in a manner that is transparent to execution of the rule program 66. For example, suppose a particular component provides an operation that looks for IP addresses for a particular sub-net, and then sends such IP addresses to a cache. If the presence of the IP address in the cache is still valid, even if the rule contains an operation that ~~subsequently fail,~~ subsequently fails, then the operation should be declared as a monitor, otherwise it is declared as an action.

Rewrite the paragraph beginning on page 25, line 21, to read as follows:

<ruleHdr>

The rule header contains information which ~~pertains to all~~ pertains to the whole rule 68.

Rewrite the paragraph beginning on page 26, line 14, to read as follows:

<keyDecl>

The key of a rule 68 ~~is hexadecimal~~ is a hexidecimal string ~~which used~~ which is used to ~~authenticating~~ authenticate the rule's origin. When the virtual machine 10 loads a rule, it ensures that the key of the rule 68 is compatible with a "shared secret" that has been assigned to the relevant

network device.

Rewrite the paragraph beginning on page 26, line 21, to read as follows:

<constant>

Constant data items are compiled into the heap-objects or inline-objects and can be ~~reforced~~ referred to by ~~use of an assigned~~ use of assigned identifiers.

Rewrite the paragraph beginning on page 27, line 3, to read as follows:

<heapObject>

A heap object is to be stored in an area of the rule 68 called the parameter heap. These items are treated as contiguous, modulo 4 sequence of bytes. The first 2 bytes of the heap object is the type field, the second 2 bytes of the heap object is the length field in bytes, and the remaining bytes are the ~~objects value~~ object's value followed possibly by padding.

Rewrite the paragraph beginning on page 33, line 2, to read as follows:

According to a further aspect of the present invention, there is provided a client application that executes on a network client device (e.g., a workstation 102) so as to allow a network connection device 12 (e.g., a switch, bridge or router) to interact with a network client device as though it were a host-coupled device. The client application provides a number of functions, which will be described below. In the exemplary embodiment described below, the client application has been conveniently labeled as a virtual network interface ~~(VNIC) client~~ client (VNIC) application 100. It will nonetheless be appreciated that this is merely a convenient label for the exemplary embodiment.

Rewrite the paragraph beginning on page 42, line 21, to read as follows:

If written in a programming language conforming to a recognized standard, the software 226 can be executed on a variety of hardware platforms and for interface to a variety of operating systems. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein. Furthermore, it is common in the art to speak of software, in one form or another (e.g., program, procedure, process, application, module, logic...), as taking an action or causing a result. Such expressions are merely a shorthand way of saying that execution of the software by a machine, such as the computer system ~~200, the~~ 200, causes the machine to perform an action or a produce a result.